Basic Processor Design

Hung-Wei Tseng
Both version A and B produce the same output. Without compiler optimization, which version of code would have significantly better performance?

A. Version A  
B. Version B  
C. They are about the same (less than 10% difference)
Both version A and B produces the same output. Without compiler optimization, which version of code would have significantly better performance?

A. Version A
B. Version B
C. They are about the same (less than 10% difference)
Outline

• Single-cycle processor
• Pipelining
• The 5-stage MIPS Pipeline processor
• Pipeline Hazards
The “life” of a dynamic instruction?

- Instruction Fetch (IF)
  - Fetch the **instruction** pointed by PC from memory
- Instruction Decode (ID)
  - Decode the instruction for the desired operation and operands
  - Reading source **register** values
- Execution (EX)
  - ALU instructions: Perform **ALU** operations
  - Conditional Branch: Determine the branch outcome (taken/not taken)
  - Memory instructions: Determine the effective address for data memory access
- Data Memory Access (MEM) — Read/write data memory
- Write Back (WB) — Present ALU result/read value in the target **register**
- Update PC
  - If the branch is taken — set to the branch target address
  - Otherwise — advance to the next instruction — current PC + 4
Single-cycle processor
Single-cycle processor

Instruction Fetch
- PC
- Instruction Memory
  - Instruction Address [31:0]
  - Instruction [31:0]

Instruction Decode, prepare operands
- Instruction [15-0]
- Instruction [15-11]
- Instruction [20-16]
- Instruction [25-21]
- Instruction [31-26]

Register File
- Read Reg 1
- Read Data 1
- Read Reg 2
- Read Data 2
- Write Reg
- Write Data
- MemToReg
- Branch
- MemoryRead
- MemoryWrite
- ALUOp
- ALUSrc
- RegWrite

ALU
- ALU Ctrl.
- ALU Op
- Is zero?
- sign-extend

Write Back
- Write Reg
- Write Data

Data Memory
- Data Address [31:0]
- Write Data[31:0]

Write Back
- Data Memory

Next PC
- Control
- Next PC

Instruction Memory
- Instruction Fetch
- Instruction Decode
- Prepare operands
- Execute
- Write Back
- Next PC
- PC
- RegDst
- MemToReg
- Branch
- MemoryRead
- MemoryWrite
- ALUOp
- ALUSrc
- RegWrite

ALU
- ALU Ctrl.
- ALU Op
- Is zero?
- sign-extend

Data Memory
- Data Address [31:0]
- Write Data[31:0]

Write Back
- Data Memory
- Write Back

Next PC
- Control
- Next PC
- Next PC
- PC
- RegDst
- MemToReg
- Branch
- MemoryRead
- MemoryWrite
- ALUOp
- ALUSrc
- RegWrite

ALU
- ALU Ctrl.
- ALU Op
- Is zero?
- sign-extend

Write Back
- Write Reg
- Write Data

Data Memory
- Data Address [31:0]
- Write Data[31:0]
Performance of a single-cycle processor

- How many of the following statements about a single-cycle processor is correct?
  
  ① The CPI of a single-cycle processor is always 1
  ② If the single-cycle implements MIPS ISA, the memory instruction will determine the cycle time
  ③ Hardware elements are mostly idle during a cycle
  ④ We can always reduce the cycle time of a single-cycle processor by supporting fewer instructions

A. 0
B. 1
C. 2
D. 3
E. 4
Performance of a single-cycle processor

- How many of the following statements about a single-cycle processor is correct?
  1. The CPI of a single-cycle processor is always 1
  2. If the single-cycle implements MIPS ISA, the memory instruction will determine the cycle time
  3. Hardware elements are mostly idle during a cycle
  4. We can always reduce the cycle time of a single-cycle processor by supporting fewer instructions

A. 0
B. 1
C. 2
D. 3
E. 4
Performance of a single-cycle processor

• How many of the following statements about a single-cycle processor is correct?
  ① The CPI of a single-cycle processor is always 1
  ② If the single-cycle implements MIPS ISA, the memory instruction will determine the cycle time
  ③ Hardware elements are mostly idle during a cycle
  ④ We can always reduce the cycle time of a single-cycle processor by supporting fewer instructions — Only if this instruction is the most time-critical one

A. 0
B. 1
C. 2
D. 3
E. 4
• Both version A and B produces the same output. On the same single-cycle processor, which version of code performs better?
A. Version A
B. Version B
C. They are about the same
Which version is faster? (2)

- Both version A and B produces the same output. On the same single-cycle processor, which version of code performs better?
  A. Version A
  B. Version B
  C. They are about the same
Which version is faster?

• Both version A and B produces the same output. On the same single-cycle processor, which version of code performs better?
  A. Version A
  B. Version B
  C. They are about the same

— The CPI of a single core processor is always 1
Pipelining
Pipelining

- Different parts of the processor works on different instructions simultaneously.
- A clock signal controls and synchronize the beginning and the end of each part of the work.
- A pipeline register between different parts of the processor to keep intermediate results necessary for the upcoming work.
Pipelining
After this point, we are completing an instruction each cycle!

All hardware parts are in use

Cycles  
Instruction  = 1

Pipelining
The following diagram shows the latency in each part of a single-cycle processor:

If we can make each part as a “pipeline stage”, what’s the maximum speedup we can achieve? (choose the closest one)

A. 3.33
B. 4
C. 5
D. 6.67
E. 10
Critical path is the longest possible delay between two registers in a design.

The critical path sets the cycle time, since the cycle time must be long enough for a signal to traverse the critical path.

Lengthening or shortening non-critical paths does not change performance.

Ideally, all paths are about the same length.
Performance of pipelining

- The following diagram shows the latency in each part of a single-cycle processor:

![Diagram showing latency in each part of a single-cycle processor]

If we can make each part as a "pipeline stage", what's the maximum speedup we can achieve? (choose the closest one)

A. 3.33
B. 4
C. 5
D. 6.67
E. 10

\[
\text{Speedup} = \frac{\text{\# of \ insts} \times 1 \times 10\text{ns}}{\text{\# of \ insts} \times 1 \times 3\text{ns}} = 3.33
\]
Limitations of pipelining

• How many of the following descriptions about pipelining is correct?
  ① You can always divide stages into short stages with latches to improve performance
  ② Pipeline registers incur overhead for each pipeline stage
  ③ The latency of executing an instruction in a pipeline processor is longer than a single-cycle processor
  ④ The throughput of a pipeline processor is usually better than a single-cycle processor

A. 0
B. 1
C. 2
D. 3
E. 4
Limitations of pipelining

- How many of the following descriptions about pipelining is correct?
  1. You can always divide stages into short stages with latches to improve performance
  2. Pipeline registers incur overhead for each pipeline stage
  3. The latency of executing an instruction in a pipeline processor is longer than a single-cycle processor
  4. The throughput of a pipeline processor is usually better than a single-cycle processor

A. 0
B. 1
C. 2
D. 3
E. 4
Limitations of pipelining

- How many of the following descriptions about pipelining is correct?
  1. You can always divide stages into short stages with latches to improve performance
  - Only if this stage is the most time-critical one
  2. Pipeline registers incur overhead for each pipeline stage
  3. The latency of executing an instruction in a pipeline processor is longer than a single-cycle processor
  - You have pipeline registers and each stage needs to be equally long
  4. The throughput of a pipeline processor is usually better than a single-cycle processor

A. 0
B. 1
C. 2
D. 3
E. 4
Performance of pipelining

- The following diagram shows the latency in each part of a single-cycle processor:

If we can make each part as a “pipeline stage”, what’s the maximum speedup we can achieve? (choose the closest one)

A. 3.33
B. 4
C. 5
D. 6.67
E. 10

\[ \text{Speedup} = \frac{\#_{\text{of\_insts}} \times 1 \times 10\text{ns}}{\#_{\text{of\_insts}} \times 1 \times 3\text{ns}} = 3.33 \]

- The cycle time is 3ns
- Each instruction now takes “15ns” to leave the pipeline!
The 5-stage MIPS Pipeline Processor
Single-cycle processor
Pipelined processor

- Instruction Memory
- Register File
- ALU
- Data Memory

**Control Flow Diagram:**
- **IF/ID:**
  - Instruction Address [31:0]
- **ID/EX:**
  - Read Reg 1
  - Read Data 1
  - Read Reg 2
  - Read Data 2
  - Write Reg
  - Write Data
- **EX/MEM:**
  - MemToReg
  - MemoryRead
  - MemoryWrite
  - ALUOp
  - ALUSrc
  - RegWrite
  - Is zero?
- **MEM/WB:**
  - Data Address [31:0]
  - Write Data [31:0]
Pipelined processor

add $1, $2, $3
lw $4, 0($5)
sub $6, $7, $8
sub $9, $10, $11
sw $1, 0($12)
add $1, $2, $3
lw $4, 0($5)
sub $6, $7, $8
sub $9, $10, $11
sw $1, 0($12)
Pipelined processor

add $1, $2, $3
lw $4, 0($5)
sub $6, $7, $8
sub $9, $10, $11
sw $1, 0($12)
Pipelined processor

add $1, $2, $3
lw $4, 0($5)
sub $6, $7, $8
sub $9,$10,$11
sw $1, 0($12)
Pipelined processor

```
add $1, $2, $3
lw $4, 0($5)
sub $6, $7, $8
sub $9, $10, $11
sw $1, 0($12)
```
Each instruction has to go through all 5 pipeline stages: IF, ID, EXE, MEM, WB in order — only valid if it’s single-issue, MIPS 5-stage pipeline.

An instruction can enter the next pipeline stage in the next cycle if:
- No other instruction is occupying the next stage.
- This instruction has completed its own work in the current stage.
- The next stage has all its inputs ready and it can retrieve those inputs.

Fetch a new instruction only if:
- We know the next PC to fetch.
- We can predict the next PC.
- Flush an instruction if the branch resolution says it’s mis-predicted.
Pipelining

```plaintext
add $1, $2, $3
lw  $4, 0($5)
sub $6, $7, $8
sub $9, $10, $11
sw  $1, 0($12)
xor $13, $14, $15
and $16, $17, $18
add $19, $20, $21
sub $22, $23, $24
lw  $25, 4($26)
sw  $27, 0($28)
```

After this point, we are completing an instruction each cycle!
Both version A and B produces the same output. Without compiler optimization, which version of code would have significantly better performance?

A. Version A
B. Version B
C. They are about the same (less than 10% difference)

— Because we have pipelined instructions, the CPI of one instruction doesn’t matter as long as we can keep the pipeline busy
Announcements

• Plan your time carefully next week!
  • Reading quiz — due tomorrow
  • Assignment #2 — due next Monday
  • Assignment #3 — due next Wednesday
  • Midterm
    • Next Thursday during the lecture — make sure that you’re available, we will use Zoom
    • The lecture on Wednesday will provide a review, highlight, sample midterm

• Resources
  • Ask questions — piazza
  • Reading quizzes, turning in assignments — Canvas
  • Slides, schedule, assignment questions — Check our website
  • Video archive — Prof. Usagi’s Youtube channel